5. Quincy

Program Name: Quincy.java Input File: quincy.dat

Quincy loves floating point numbers, especially the classic irrational values like PI and E, because they go on forever! He has found some limited precision representations of these numbers, as well as several others with precision values of over 1000 places, and has decided to do some analysis research on them. He wants to know how many instances of each digit there are in these values, and then will represent those counts in a histogram. He also wants to know how the frequency changes when the values are halved or doubled. He starts with some easy values, just to test his process for accuracy, and then tries it on the longer ones.

For example, with the value 123456.7089, it is easy to see one instance of each digit. If he halves this value to get 61728.35445, the digit frequencies change a bit, and if he doubles the value, they change again. He decides to output all three versions of the value - the original, half the value, and twice the value - and then calculates and displays the digit frequency for each one. Below are several results of his program. By the number of stars, you can see one zero digit in the original number, but none in the half value or double value. The number of 2s, 3s, 6s, 7s, and 8s remain the same, and the frequencies for the rest of the digits change a bit.

```
123456.7089
61728.35445
246913.4178
0 *||
1 *|*|**
2 *|*|*
3 *|*|*
4 *|**|**
5 *|**|
6 *|*|*
7 *|*|*
8 *|*|*
9 *||*
```

Then he found a limited precision version of PI, containing 50 digits, and processed that value the same way, with the following results.

```
3.14159265358979311599796346854418516159057617187 5
1.57079632679489655799898173427209258079528808593 75
6.28318530717958623199592693708837032318115234375 0
0 *|****|****
1 *********|**|******
2 *|*****|*****
3 ****|***|*********
4 ****|**|*
5 *********|*******|*****
6 *****|***|***
7 *****|********|*****
8 ****|*******|******
9 *******|*********|*****
```

After searching some more, he found a version of PI with over 200 digits, and decided to modify the histogram output, showing up to 25 stars for each count, or just a value followed by "*s" for any counts beyond 25, so that his output would stay on the screen and not wraparound. He also limited the value output to no more than 60 digits for the same reason. Here is the outcome for that more precise value of PI.

```
3.14159265358979323846264338327950288419716939937510582097 49
1.57079632679489661923132169163975144209858469968755291048 74
6.28318530717958647692528676655900576839433879875021164194 98
0 ********************|********************|*****************
1 **********************|26*s|************************
2 ************************|************************|************************
3 ********************|******************|****************
4 ************************|27*s|******************
5 **********************|**********************|********************
6 ******************|********************|29*s
7 *************|*********************|******************
8 27*s|*************|***********************
9 ************************|************************|************************
```

Write a program that will show Quincy's process as described above.

Input - several floating point values, each on one line, each containing at least 3 digits (including the .) and no more than 1000 digits.

Output - Three versions of each value - the original value, half the value, and twice the value - showing no more than 60 digits of each, followed by a histogram showing the digit frequency counts as described and shown above.

Sample input:

```
123456.7089
3.14159265358979311599796346854418516159057617187 5
3.14159265358979323846264338327950288419716939937510582097494459 (...over 200 total digits)
```

Sample output:

See output examples above. One blank line will follow each output set.